CS 4476: Computer Vision, Fall 2020

PS0

Instructor: Devi Parikh

Due: Wednesday, August 26th, 11:59:00 pm

Instructions

1. The deliverables for this assignment must be submitted on Gradescope. Please follow the submission instructions carefully.
   - Save your written answers in a file named: lastname_firstname_ps0.pdf. LaTeX is strongly encouraged.
   - Answers to each problem must be on a separate page (or pages). On Gradescope uniquely assign each page to a single problem. Multiple sub-problems can be included on the same page.
   - Upload your pdf to the `PS0` assignment on Gradescope.
   - Save your code and input/output images in a zip file named: lastname_firstname_ps0.zip.
   - Be sure to include all the files listed in the submission checklist.
   - Upload your zip file to the `PS0 Code` assignment on Gradescope.
2. For the implementation questions, make sure your code is bug-free and works out of the box. Please be sure to submit all main and helper functions. Be sure to not include absolute paths. Points will be deducted if your code does not run out of the box.
3. If plots are required, you must include them in your answer sheet (pdf) and your code must display them when run. Points will be deducted for not following this protocol.

Getting Started

If you are not familiar with Python and Numpy read through one of the following resources:

- http://cs231n.github.io/python-numpy-tutorial/
- https://www.pythonlikeyoumeanit.com/

Note: please use Python 3.7+ for all assignments in this course.

Short answer problems [64 points]

1. [18 points] Describe (in words) the result of each of the following Python commands. Search the Numpy API documentation https://numpy.org/doc/stable/ if needed, but try to determine the output without entering the commands into Python. *Note: do not submit a screenshot of the result of typing these commands.*

```
>>> import numpy as np
```

(a)
```
>>> x = np.random.permutation(1000)
```

(b)
```
>>> a = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>> b = a[2,:]
```

(c)
```
>>> a = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>> b = a.reshape(-1)
```

(d)
```
>>> x = np.random.randn(5,1)
>>> y = x[x>0]
```

(e)
```
>>> x = np.zeros(10)+0.5
>>> y = 0.5*np.ones(len(x))
>>> z = x + y
```

(f)
```
>>> a = np.arange(1,100)
>>> b = a[::-1]
```

2. [16 points] Write functions for each of the following operations. Copy and paste your code into the answer sheet AND submit the code in a file called `q2.py`. *Note: make sure you use the numpy methods if they are specified in the question.*

   (a) Write a function `random_dice(N)` that returns the roll of a six-sided die over `N` trials using `np.random.rand`.

   (b) Write a function `reshape_vector(y)` that takes a six element vector (e.g. `y = np.array([1, 2, 3, 4, 5, 6])`) and converts it into a two column matrix (e.g. `np.array([[1,2],[3,4],[5,6]])`) using `np.reshape`.

   (c) Write a function `max_value(z)` that takes a 2D matrix (e.g. `z = np.array([[1,2],[3,4],[5,6]])`) and returns the row and column of the first occurrence of the maximum value using `np.max` and `np.where`.

   (d) Write a function `count_ones(v)` that returns the number of 1's that occur in the vector `v`.

3. [30 points] Create any 100 x 100 matrix `A` (not all constant). Save `A` as `q3-input.npy` and submit the file. Write a script named `q3.py` that loads `q3-input.npy` and performs each of the following operations on `A`. Include the script `q3.py` in your submission.

   (a) Sort the intensity values of `A` in decreasing order (ignoring the 2D structure of `A`).
   - Save the sorted intensities as `q3-output-sorted.npy` and submit the file.
   - Plot the sorted intensities in your script.
   - Display the plot in your answer sheet.

   (b) Plot a histogram of the intensity values of `A` using 20 bins (again, ignoring the 2D structure).
   - Plot the histogram in your script.
   - Display the histogram in your answer sheet.

   (c) Create a new matrix `X` that consists of the bottom left quadrant of `A`.
   - Save `X` as `q3-output-x.npy` and submit the file.

- Plot `X` in your script using `matplotlib.pyplot.imshow`.
  - *Make sure there is no interpolation (blurry effect). See the docs for* `matplotlib.pyplot.imshow`.
- Display `X` as an image in your answer sheet.

(d) Create a new matrix `Y`, which is the same as `A`, but with `A`'s mean intensity value subtracted from each pixel.

- Save `Y` as `q3-output-y.npy` and submit the file.
- Plot `Y` in your script using `matplotlib.pyplot.imshow`.
- Display `Y` as an image in your answer sheet.

(e) Create a new matrix `Z` that represents a color image the same size as `A`, but with 3 channels to represent R, G and B values. Set the values to be red (i.e., R = 1, G = 0, B = 0) wherever the intensity in `A` is greater than the average intensity in `A`, and black everywhere else.

- Save `Z` as an image `q3-output-z.png` and submit the file.
  - *Be sure to open the* ***`png`*** *in a regular image viewer to make sure it looks right.*
- Plot `Z` in your script using `matplotlib.pyplot.imshow`.
- Display `Z` as an image in your answer sheet.

4 Short programming problem [36 points]

Choose any color image from the web or your personal collection and save it as `q4-input.png`. Write a script named `q4.py`, which performs the following transformations and displays the results in a figure using `matplotlib.pyplot.subplots` in a 3x2 grid (3 rows and 2 columns). Each subplot should contain the output of each of the below operations. Label each subplot with an appropriate `title`. Provide the subplot in your answer sheet. Avoid using loops.

Note: All of the transformed output images should be save in a ***`png`*** *format.*

(a) Load the input color image and swap its red and green color channels.

- Save the output as `q4-output-swapped.png`.

(b) Convert the input color image to a grayscale image.

- Save the output as `q4-output-grayscale.png`.

(c) Convert the grayscale image to its negative image, in which the lightest values appear dark and vice versa.

- Save the output as `q4-output-negative.png`.

(d) Map the grayscale image to its mirror image, i.e., flipping it left to right.

- Save the output as `q4-output-mirror.png`.

(e) Average the grayscale image with its mirror image (use typecasting).

- Save the output as `q4-output-average.png`.

(f) Create a matrix `N` whose size is same as the grayscale image, containing random numbers in the range [0 255]. Add `N` to the grayscale image, then clip the resulting image to have a maximum value of 255.

- Save the noise matrix as `q4-noise.npy`.
- Save the output as `q4-output-noise.png`.

Be sure to submit the input image, all the output images, and `noise.npy`.

Tips:

- Do the necessary typecasting (`uint8` and `double`) when working with or displaying the images.
- If you can't find some functions in numpy (such as rgb2gray), you can write your own function. For example:

```
def rgb2gray(rgb):
  return np.dot(rgb[...,:3], [0.2989, 0.5870, 0.1140])
```
[1]

Submission Checklist

☐ Answer sheet containing your responses to problems 1-4 named `LastName_FirstName_PS0.pdf` (minimum 4 pages).

☐ `Zip` file named `LastName_FirstName_PS0.zip` containing:

- ☐ `q2.py`
- ☐ `q3.py`
- ☐ `q3-input.npy`
- ☐ `q3-output-sorted.npy`
- ☐ `q3-output-x.npy`
- ☐ `q3-output-y.npy`
- ☐ `q3-output-z.png`
- ☐ `q4.py`
- ☐ `q4-input.png`
- ☐ `q4-noise.npy`
- ☐ `q4-output-swapped.png`
- ☐ `q4-output-grayscale.png`
- ☐ `q4-output-negative.png`
- ☐ `q4-output-mirror.png`
- ☐ `q4-output-average.png`
- ☐ `q4-output-noise.png`

[1] These are the weights that the MATLAB rbg2gray function uses.
This assignment closely follows the PS0 assignment of Kristen Grauman's CS 376: Computer Vision at UT Austin